Práctica 1 Aprendizaje Automático

Antonio Álvarez Caballero

Generación y visualización de datos

Ejercicio 1

En primer lugar, creamos una función que crea un data.frame con valores aleatorios según una distribución uniforme. *N* es el número de filas del data.frame, *dim* el número de columnas y *rango* el rango donde estarán los valores.

```
simula_unif <- function(N, dim, rango){
  res <- data.frame(matrix(nrow = N, ncol = dim))

  for(i in 1:N){
    res[i,] <- runif(dim, rango[1], rango[length(rango)])
  }

  names(res) <- c("X", "Y")

  return(res)
}
```

Ejercicio 2

Del mismo modo creamos una función análoga para la distribución *normal* o *gaussiana*.

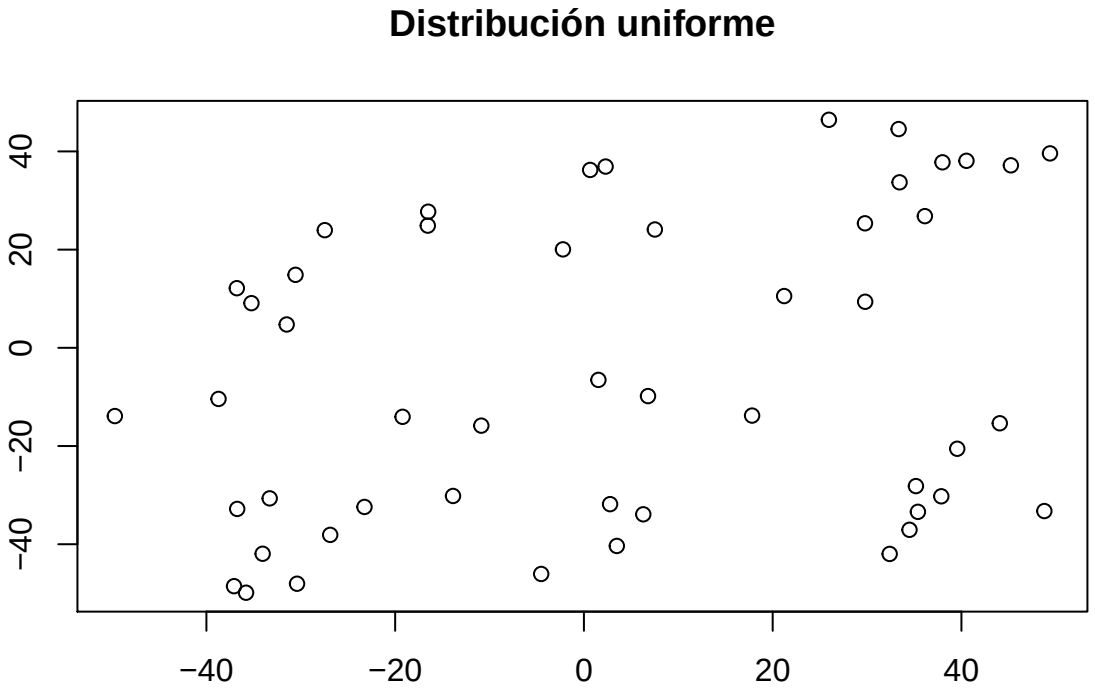
```
simula_gauss <- function(N, dim, sigma){
  res <- data.frame(matrix(nrow = N, ncol = dim))

  for(i in 1:N){
    res[i,] <- rnorm(dim, sd = sqrt(sigma))
  }

  names(res) <- c("X", "Y")

  return(res)
}
```

Ejercicios 3 y 4

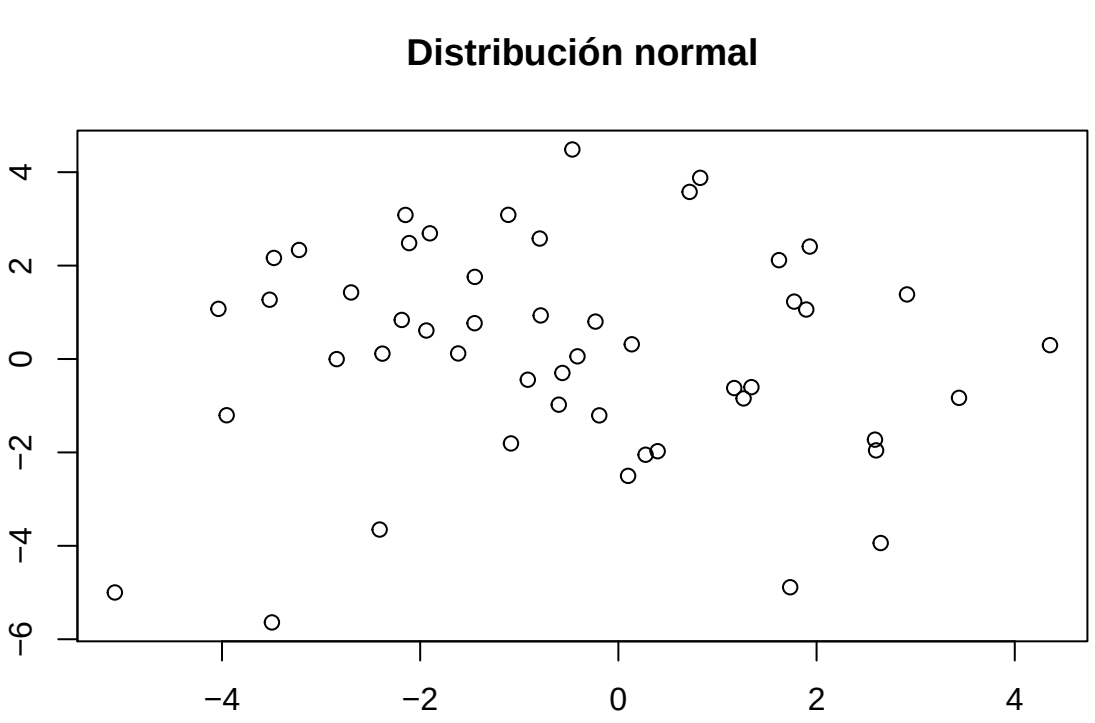
Ahora asignamos los resultados a sendos objetos del tipo *data.frame* y las dibujamos.

```
muestra.uniforme <- simula_unif(50, 2, -50:50)
muestra.gaussiana <- simula_gauss(50, 2, 5:7)
```

```r
plot(muestra.uniforme, main = "Distribución uniforme", xlab = "", ylab = "")
```




```r
plot(muestra.gaussiana, main = "Distribución normal", xlab = "", ylab = "")
```



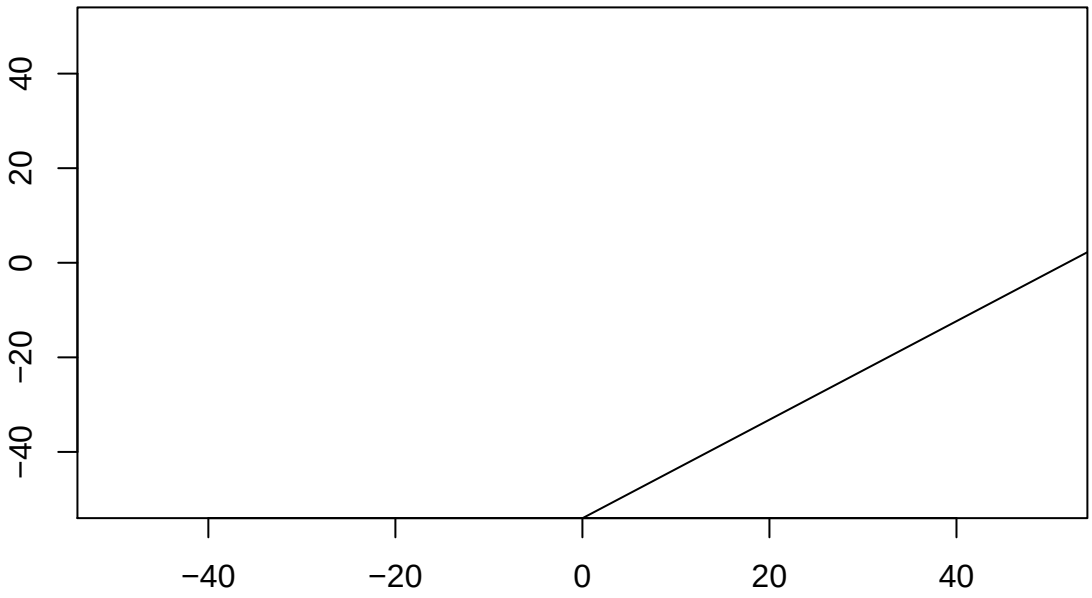
Ejercicio 5

Ahora escribimos una pequeña función para calular una recta dados dos puntos. Daremos la recta con la ecuación punto pendiente, por lo que tenemos que calcular la pendiente y la desviación.

```
simulaRecta <- function(intervalo){
  A <- runif(2, intervalo[1], intervalo[length(intervalo)])
  B <- runif(2, intervalo[1], intervalo[length(intervalo)])

  m <- (A[2] - B[2]) / (A[1] - B[1])
  b <- A[2] - m * A[1]

  return(c(b,m))
}
```

Generamos una recta aleatoria en $[-50, 50]$

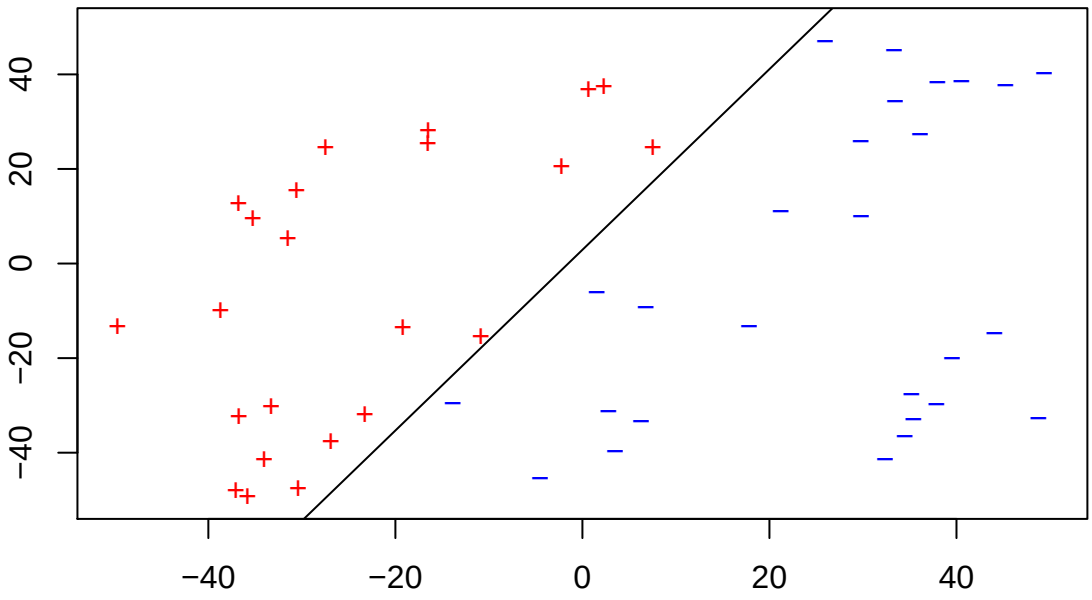
```
rectaPrueba <- simulaRecta(-50:50)

plot(1, type="n", xlab="", ylab="", xlim=c(-50, 50), ylim=c(-50, 50))
abline(coef = rectaPrueba)
```



Ejercicio 6

Ahora clasificamos los datos de la muestra uniforme según otra recta aleatoria. Guardamos las muestras clasificadas para próximos ejercicios.

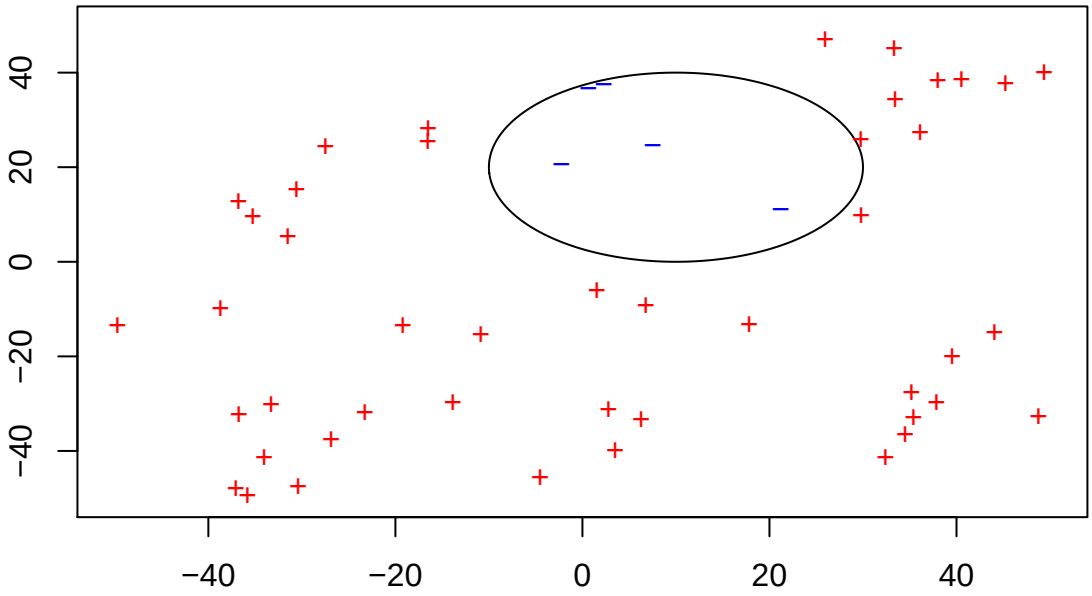
```
rectaClasificacion <- simulaRecta(-50:50)

plot(1, type = "n", xlab = "", ylab = "", xlim = c(-50, 50), ylim = c(-50,
    50))

muestra.uniforme.positiva <- subset(muestra.uniforme, Y - rectaClasificacion[2] *
    X - rectaClasificacion[1] > 0)

muestra.uniforme.negativa <- subset(muestra.uniforme, Y - rectaClasificacion[2] *
    X - rectaClasificacion[1] <= 0)

points(muestra.uniforme.positiva, pch = "+", col = "red")
points(muestra.uniforme.negativa, pch = "-", col = "blue")

abline(coef = rectaClasificacion)
```



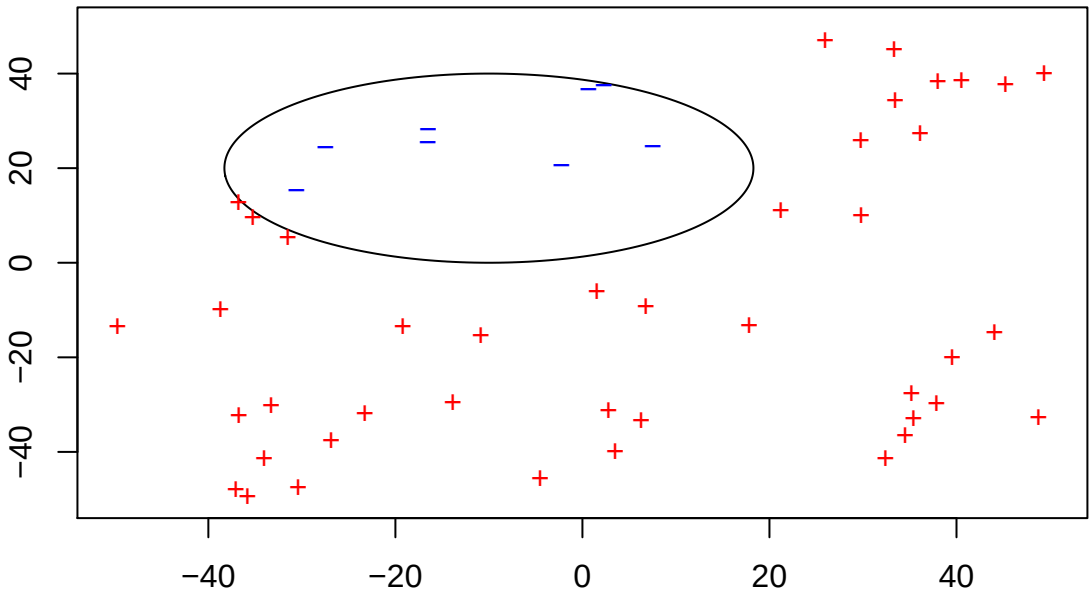
Ejercicio 7

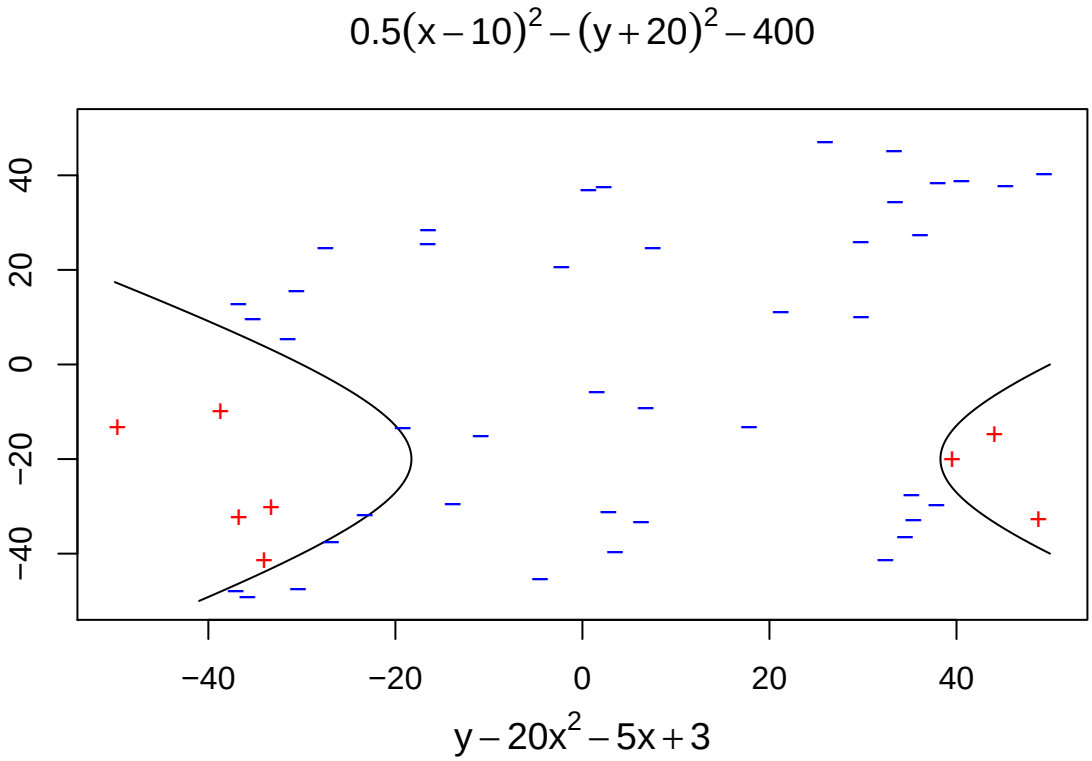
Una vez hemos clasificado los datos con una recta, clasificamos con otro tipo de funciones.

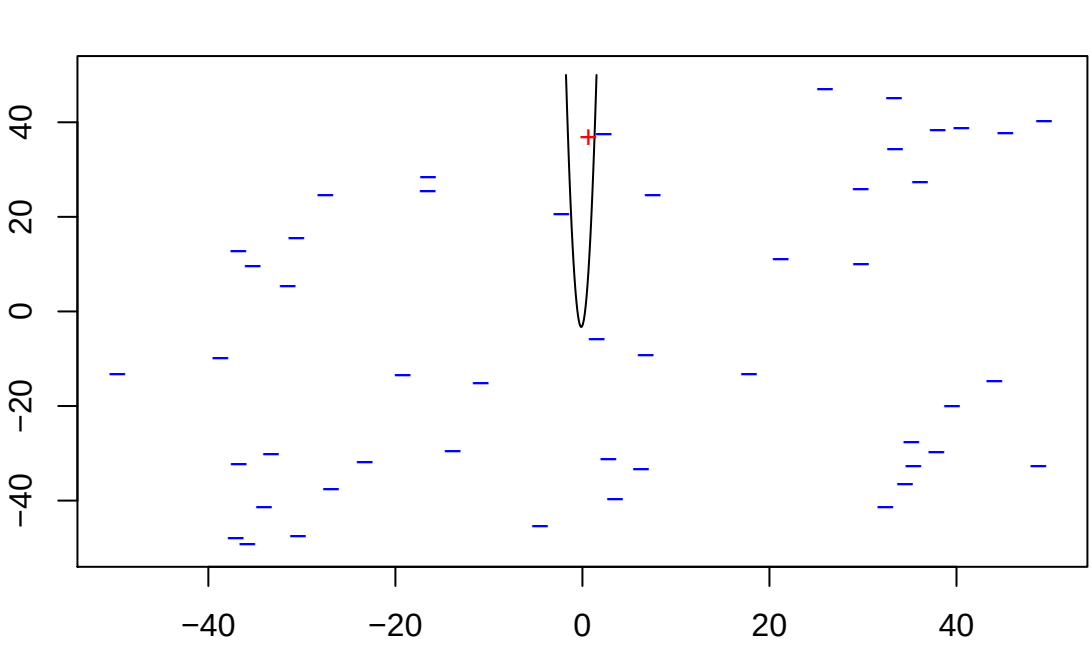
$$(x-10)^2+(y-20)^2-400$$



$$0.5(x+10)^2+(y-20)^2-400$$







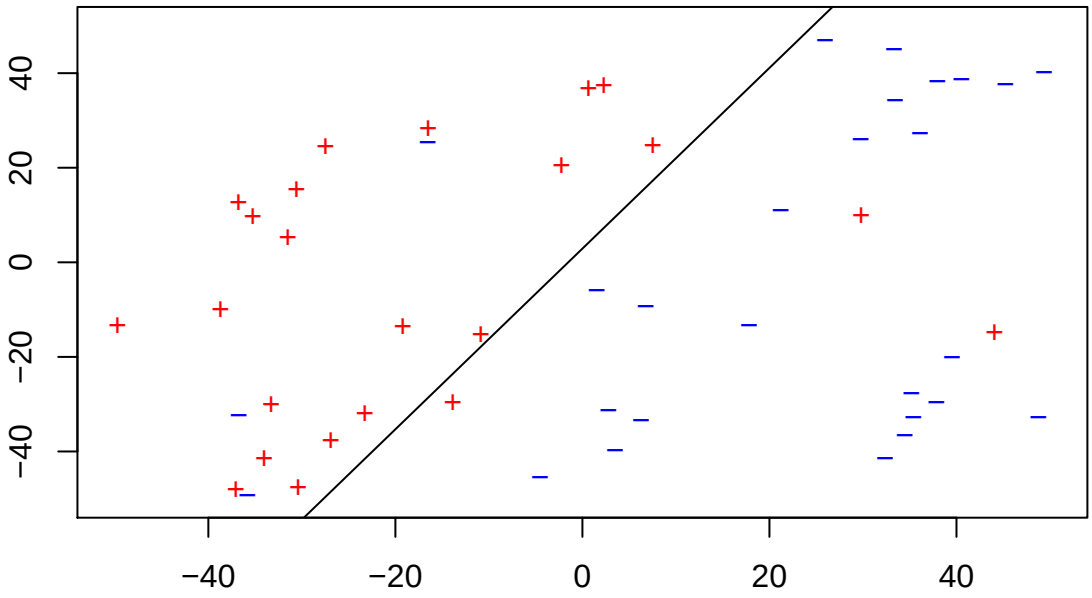
Añadir aquí conclusiones de las regiones nuevas comparadas con la lineal.

Ejercicio 8

Tomamos la muestra clasificada y creamos un vector de booleanos con el 10% *TRUE* y el restante *FALSE*. Así luego extraeremos una porción de la muestra.

```
# Tomamos la porción deseada
porcion <- 10
numero.datos.positivos <- ceiling(nrow(muestra.uniforme.positiva) * porcion/100)
numero.datos.negativos <- ceiling(nrow(muestra.uniforme.negativa) * porcion/100)


vector.aleatorio.muestras.positivas <- c(rep(FALSE, numero.datos.positivos),
    rep(TRUE, nrow(muestra.uniforme.positiva) - numero.datos.positivos))

vector.aleatorio.muestras.negativas <- c(rep(FALSE, numero.datos.negativos),
    rep(TRUE, nrow(muestra.uniforme.negativa) - numero.datos.negativos))

vector.aleatorio.muestras.positivas <- sample(vector.aleatorio.muestras.positivas)
vector.aleatorio.muestras.negativas <- sample(vector.aleatorio.muestras.negativas)

muestra.uniforme.positiva.nueva <- muestra.uniforme.positiva[vector.aleatorio.muestras.positivas,
    ]
muestra.uniforme.negativa.nueva <- muestra.uniforme.negativa[vector.aleatorio.muestras.negativas,
    ]

muestra.uniforme.positiva.nueva <- rbind(muestra.uniforme.positiva.nueva,
    muestra.uniforme.negativa[!vector.aleatorio.muestras.negativas, ])

muestra.uniforme.negativa.nueva <- rbind(muestra.uniforme.negativa.nueva,
    muestra.uniforme.positiva[!vector.aleatorio.muestras.positivas, ])
```

Ahora mostramos los nuevos resultados



Ajuste del algoritmo Perceptron (PLA)

Ejercicio 1

Vamos a escribir una función que implemente el algoritmo *Perceptron* o *PLA*. *Datos* es un *data.frame* con cada muestra por fila, *label* el vector de etiquetas, el cual debe tener la misma longitud que los datos de muestra, *max_iter* es el número máximo de iteraciones del algoritmo y *vini* es el vector inicial, que debe tener la misma dimensión que los datos.

```
ajusta_PLA <- function(datos, label, max_iter, vini){
  cambio <- TRUE
  w <- vini
  iteraciones <- 0

  while (cambio && iteraciones < max_iter){
    cambio <- FALSE
    for (i in 1:nrow(datos)){
      x_i <- c(1,as.numeric(datos[i,]))
      prodEscalar <- sum(t(w) * x_i)
      if (sign(prodEscalar) != label[i]){
        cambio <- TRUE
        w <- w + label[i] * x_i
        break
      }
    }
    iteraciones <- iteraciones + 1
  }
  print(cambio)

  resultado <- list("pesos" = w, "iteraciones" = iteraciones)
  return (resultado)
}
```

Ejercicio 2

Vamos a hacer una simulación de prueba. Tomamos los datos de la muestra uniforme y un vector de etiquetas aleatorio.

```
muestra.uniforme.etiquetada <- rbind(cbind(muestra.uniforme.positiva, etiqueta = 1),
                                     cbind(muestra.uniforme.negativa, etiqueta = -1))

rownames(muestra.uniforme.etiquetada) <- NULL

etiquetas <- t(as.vector(muestra.uniforme.etiquetada["etiqueta"]))
vini <- rep(0,3)

resultado <- ajusta_PLA(muestra.uniforme, etiquetas, 20000, vini)
```

```
## [1] TRUE
```

```r
w <- resultado$pesos
iteraciones <- resultado$iteraciones

print(w)
```

```
## [1] 1374.000000   -5.460548  -23.891026
```

```r
print(iteraciones)
```

```
## [1] 20000
```

```r
plot(1, type="n", xlab="", ylab="", xlim=c(-50, 50), ylim=c(-50, 50))


abline(coef = c(-w[1]/w[3], w[2]/w[3]))

points(subset(muestra.uniforme.etiquetada, etiqueta == -1), pch = "-", col = "blue")
points(subset(muestra.uniforme.etiquetada, etiqueta == 1), pch = "+", col = "red")
```